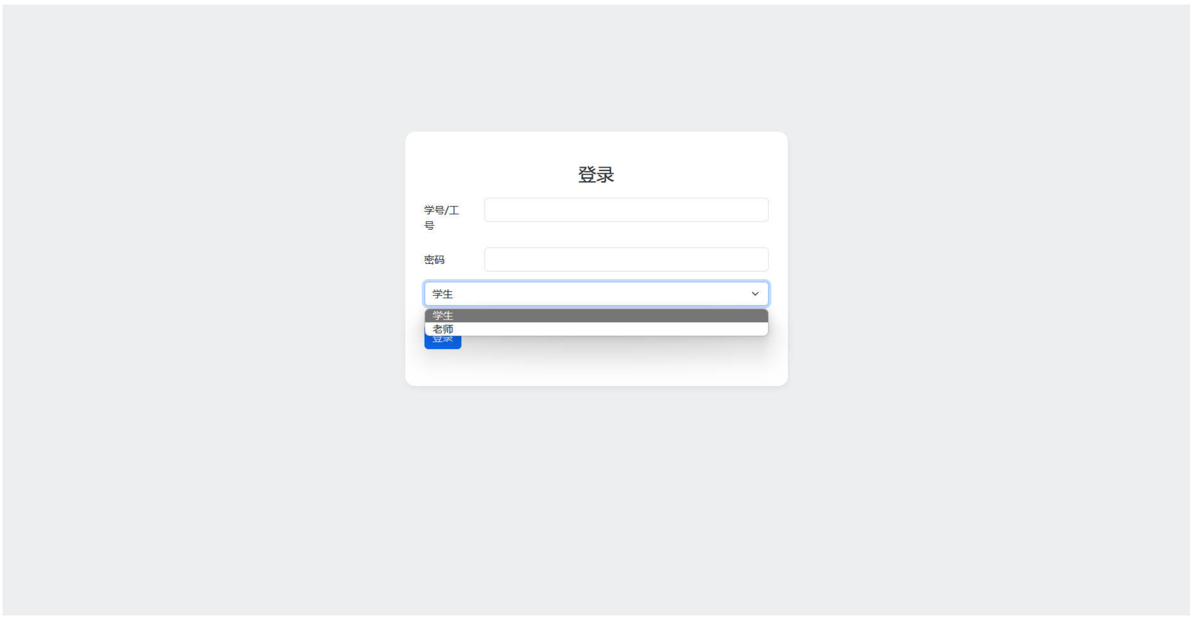
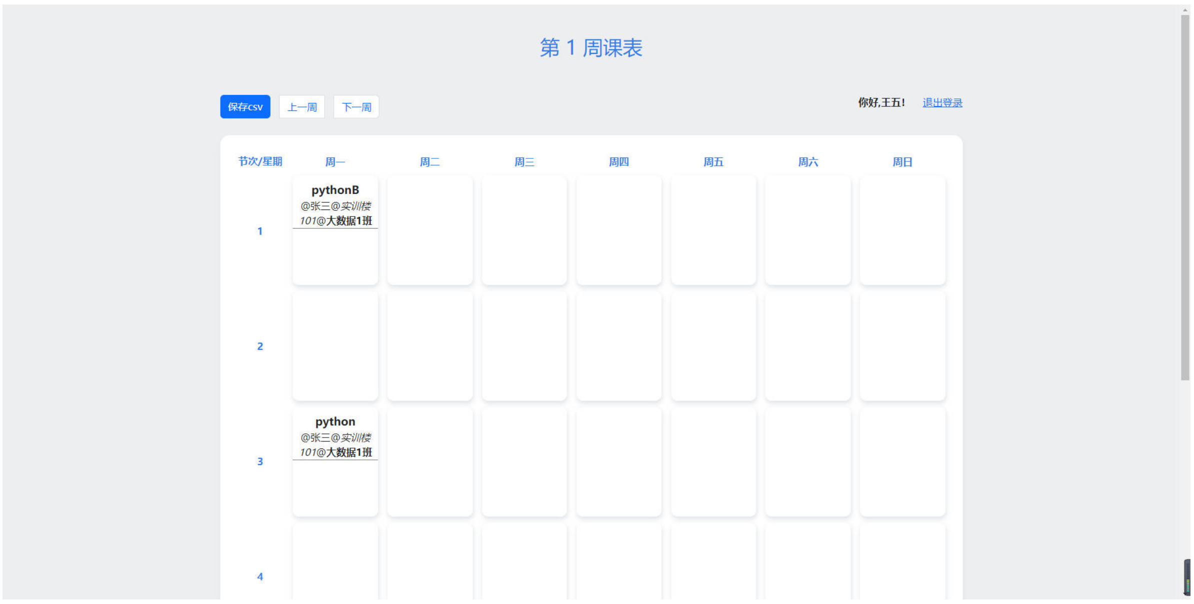
**课表管理系统**

# 课表管理系统

## 1.学生/教师用户登录界面



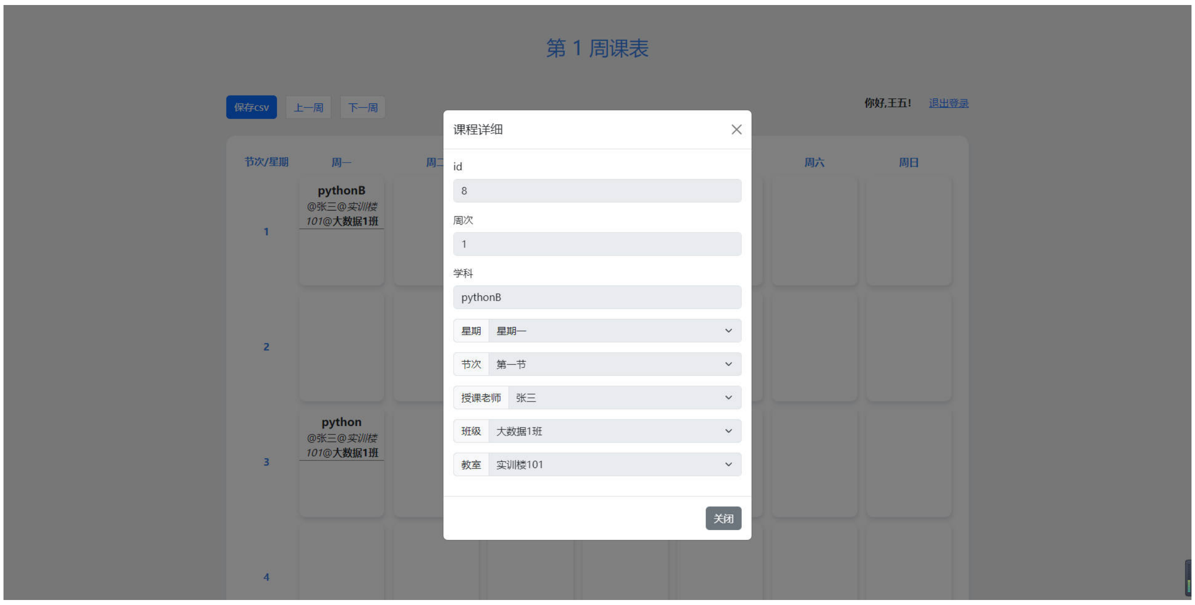
## 2.首页



| 节次/星期 | 周一 | 周二 | 周三 | 周四 | 周五 | 周六 | 周日 |
|---|---|---|---|---|---|---|---|
| 1 | pythonB @张三@实训楼101@大数据1班 | | | | | | |
| 2 | | | | | | | |
| 3 | python @张三@实训楼101@大数据1班 | | | | | | |
| 4 | | | | | | | |

## 3.课程详细

点击已有课程，可弹出课程详细数据。



第 1 周课表

课程详细

id: 8

周次: 1

学科: pythonB

星期: 星期一

节次: 第一节

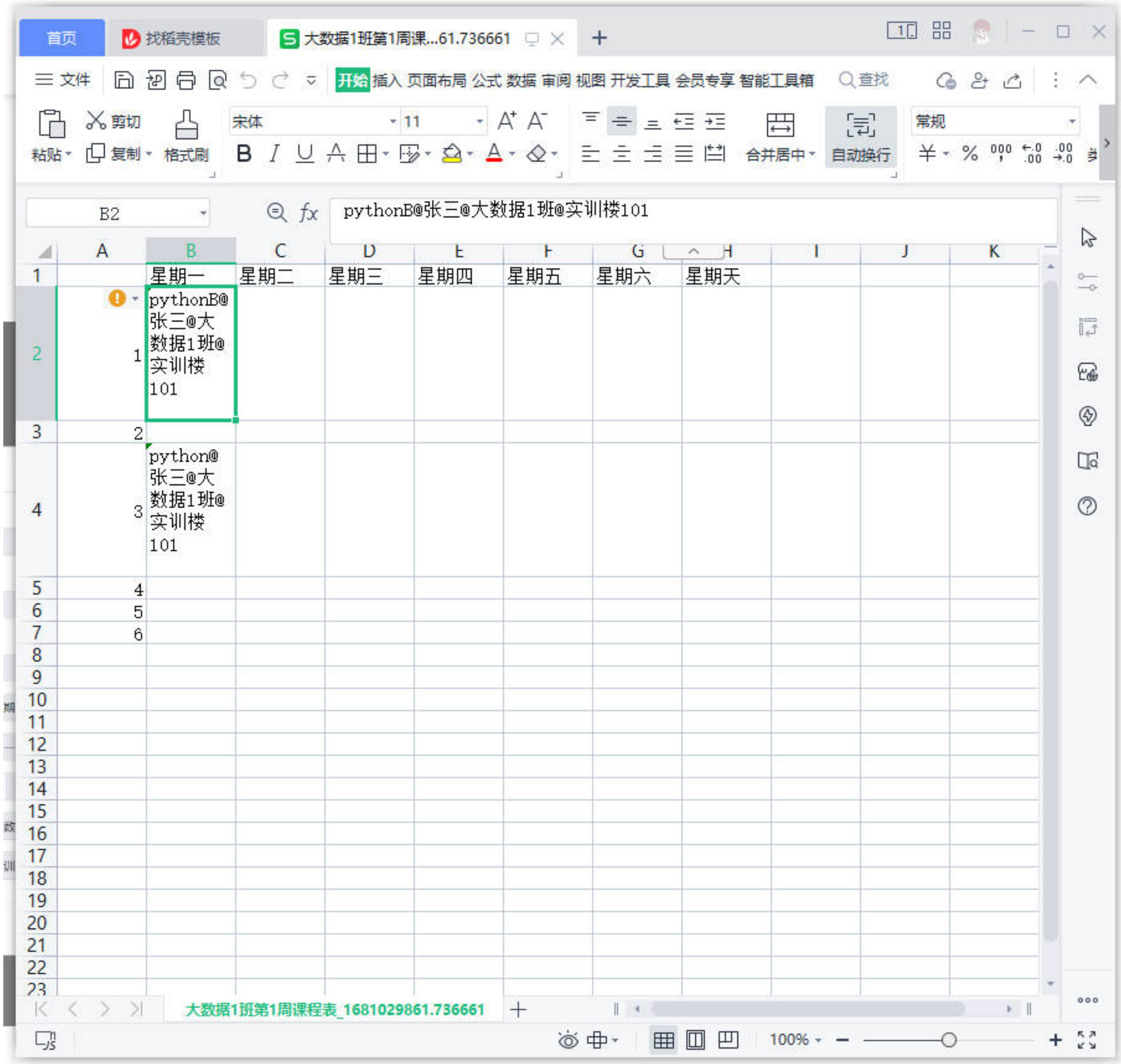
授课老师: 张三

班级: 大数据1班

教室: 实训楼101

## 4.保存csv功能

点击保存csv按钮，可下载当前显示课表的csv格式文档。

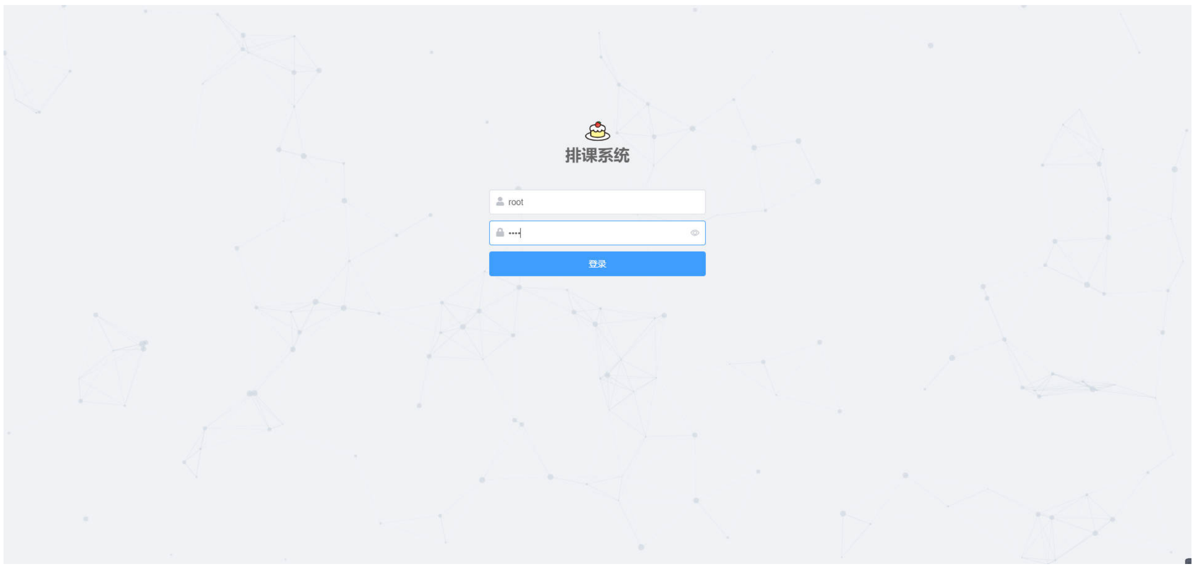


B2 fx pythonB@张三@大数据1班@实训楼101

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | | 星期一 | 星期二 | 星期三 | 星期四 | 星期五 | 星期六 | 星期天 |
| 2 | 1 | pythonB@张三@大数据1班@实训楼101 | | | | | | |
| 3 | 2 | | | | | | | |
| 4 | 3 | python@张三@大数据1班@实训楼101 | | | | | | |
| 5 | 4 | | | | | | | |
| 6 | 5 | | | | | | | |
| 7 | 6 | | | | | | | |

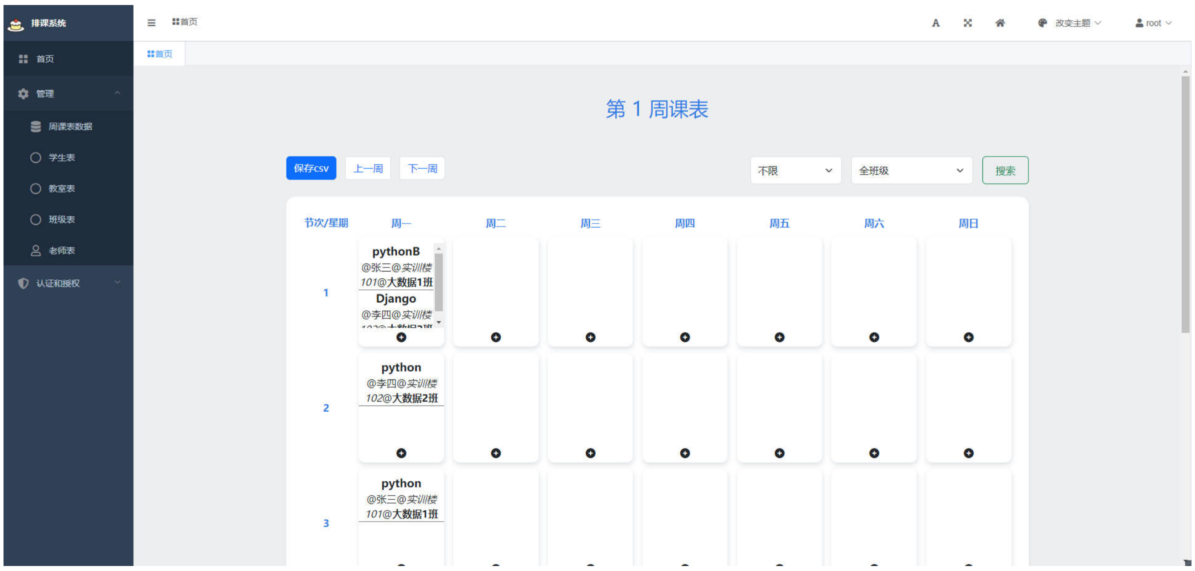
首页
找稻壳模板
大数据1班第1周课...61.736661
文件
开始 插入 页面布局 公式 数据 审阅 视图 开发工具 会员专享 智能工具箱
查找
粘贴
剪切
复制
格式刷
宋体
11
合并居中
自动换行
常规
大数据1班第1周课程表_1681029861.736661
100%

## 5.管理员登录界面



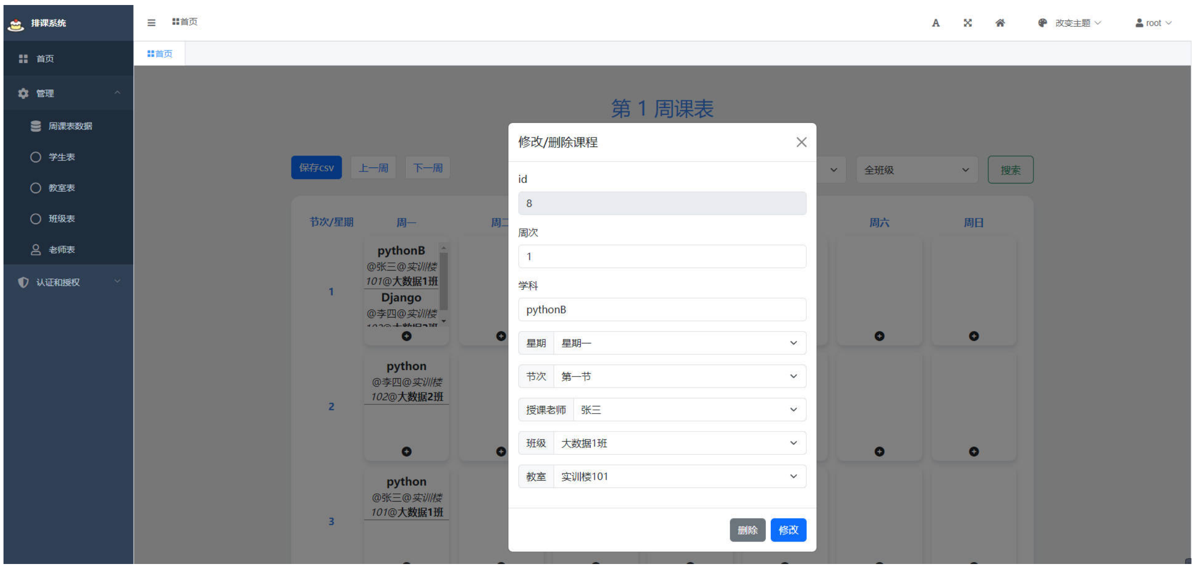
## 6.管理员后台界面

管理员通过后台可以对老师、学生、班级等数据表进行增删改查功能，同时管理员后台首页可以实时的对每个班级/老师的课程进行安排，还有简单的筛选搜索功能、导出csv功能、新增/删除/修改课表信息功能。



第 1 周课表

| 节次/星期 | 周一 | 周二 | 周三 | 周四 | 周五 | 周六 | 周日 |
|---|---|---|---|---|---|---|---|
| 1 | pythonB @张三@实训楼101@大数据1班 Django @李四@实训楼 | | | | | | |
| 2 | python @李四@实训楼102@大数据2班 | | | | | | |
| 3 | python @张三@实训楼101@大数据1班 | | | | | | |



第 1 周课表

修改/删除课程

id: 8

周次: 1

学科: pythonB

星期: 星期一

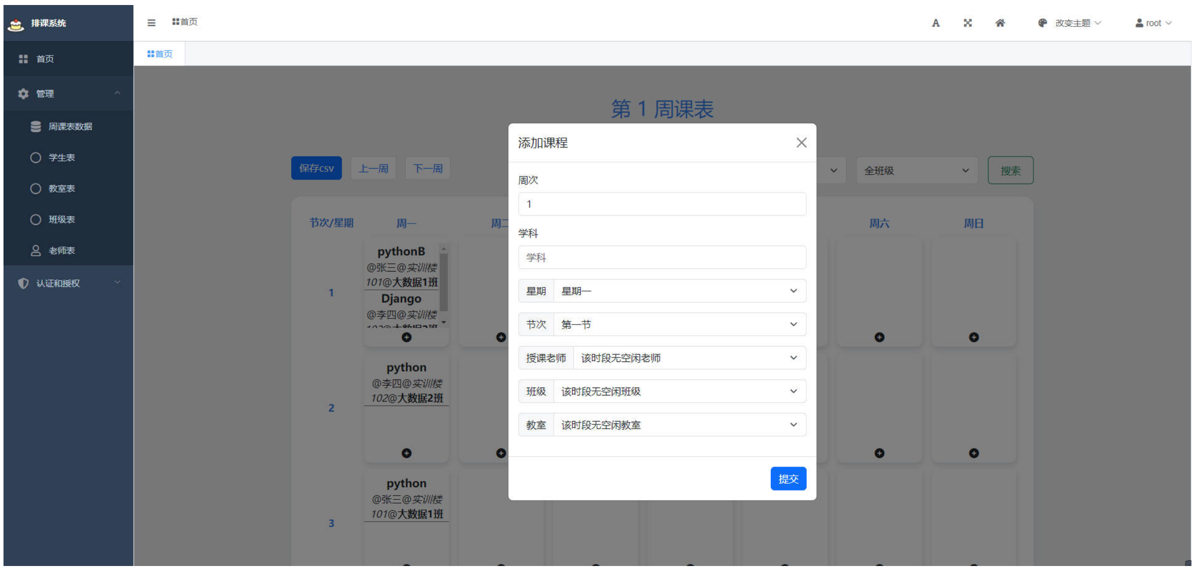
节次: 第一节

授课老师: 张三

班级: 大数据1班

教室: 实训楼101

删除 修改



第 1 周课表

添加课程

周次: 1

学科: 学科

星期: 星期一

节次: 第一节

授课老师: 该时段无空闲老师

班级: 该时段无空闲班级

教室: 该时段无空闲教室

提交

节次/星期 | 周一 | 周二 | 周六 | 周日

1: pythonB @张三@实训楼 101@大数据1班; Django @李四@实训楼

2: python @李四@实训楼 102@大数据2班

3: python @张三@实训楼 101@大数据1班

## 7.代码详解

- **数据库建模**

根据需求分析，总共建立了老师表、班级表、学生表、教室表、课程总表

- **表之间的关系**

学生表和班级表是一对多的关系，

课程表和老师表是一对多的关系，

课程表班级表是一对多的关系，

课程表教室表是一对多的关系。

```
#models.py
from django.db import models
from django.utils import timezone
# Create your models here.


#老师表
class Teacher(models.Model):
    id=models.BigAutoField(primary_key=True,blank=False)
    teacher_name = models.CharField(max_length=255,verbose_name='教师姓名')
    teacher_num = models.CharField(max_length=255, blank=True,
null=True,verbose_name='教师工号')
    password=models.CharField(max_length=200,verbose_name='密码')
    teacher_level = models.IntegerField(verbose_name='教师等级')
    sex = models.IntegerField(verbose_name='性别',help_text='1 男 2 女')
    age = models.IntegerField(blank=True, null=True,verbose_name='年龄')
    phone = models.CharField(max_length=255,verbose_name='手机号')
    birthday = models.CharField(max_length=255, blank=True,
null=True,verbose_name='出生年月')
    email = models.CharField(max_length=255, blank=True,
null=True,verbose_name='教师邮箱')
    school = models.CharField(max_length=255, blank=True,
null=True,verbose_name='毕业院校')
    department = models.CharField(max_length=255, blank=True,
null=True,verbose_name='毕业院校院系')
```

```
    major = models.CharField(max_length=255, blank=True,
null=True,verbose_name='毕业院校专业')
    education = models.CharField(max_length=255, blank=True,
null=True,verbose_name='学历')
    def __str__(self):
        return self.teacher_name
    class Meta:
        verbose_name_plural='老师表'
        # managed = False
        db_table = 'teacher'
#班级表
class BasicClass(models.Model):
    id=models.BigAutoField(primary_key=True,blank=False)
    class_name = models.CharField(max_length=200,verbose_name='班级名')
    class_cate = models.IntegerField(blank=True, null=True,verbose_name='班级分
类',help_text='0 五天全日制 1 六天全日制 2预科班 3 周末班')
    start_time = models.DateTimeField(verbose_name='开班时间')
    class_status = models.IntegerField(blank=True, null=True,verbose_name='班级状
态',help_text='0 正常 1禁用')
    description = models.CharField(max_length=200, blank=True,
null=True,verbose_name='备注')
    def __str__(self):
        return self.class_name
    class Meta:
        verbose_name_plural='班级表'
        # managed = False
        db_table = 'basic_class'

#学生表
class Student(models.Model):
    id=models.BigAutoField(primary_key=True,blank=False)
    student_name = models.CharField(max_length=200,verbose_name='学生姓名')
    password=models.CharField(max_length=200,verbose_name='密码')
    student_num = models.CharField(max_length=255, null=False,verbose_name='学
号')
    #学生表和班级表是一对多的关系，这里建立外键
    Class =models.ForeignKey(BasicClass, verbose_name='班级',
on_delete=models.CASCADE)
    add_time = models.DateTimeField(verbose_name='加入班级时间')
    student_status = models.IntegerField(verbose_name='学生状态',help_text='0正常 1
请假 2 休学 3 退学')
    sex = models.IntegerField(blank=True, null=True,verbose_name='性别')
    age = models.IntegerField(blank=True, null=True,verbose_name='年龄')
    birthday = models.CharField(max_length=200, blank=True,
null=True,verbose_name='出生年月日')
    student_email = models.CharField(max_length=200, blank=True,
null=True,verbose_name='邮箱')
    student_school = models.CharField(max_length=200, blank=True,
null=True,verbose_name='院校')
    student_department = models.CharField(max_length=200, blank=True,
null=True,verbose_name='院系')
    student_major = models.CharField(max_length=200, blank=True,
null=True,verbose_name='专业')
    student_school_class = models.CharField(max_length=200, blank=True,
null=True,verbose_name='在校班级')
```

```
    student_education = models.CharField(max_length=200, blank=True,
null=True,verbose_name='学历')
    phone = models.CharField(max_length=200,verbose_name='手机号')
    qq_number = models.CharField(max_length=200, blank=True,
null=True,verbose_name='qq号')
    wechart_number = models.CharField(max_length=200, blank=True,
null=True,verbose_name='微信号')
    idcard = models.CharField(max_length=200, blank=True,
null=True,verbose_name='身份证号')
    emergency_name = models.CharField(max_length=200, blank=True,
null=True,verbose_name='紧急联系人姓名')
    emergency_phone = models.CharField(max_length=200, blank=True,
null=True,verbose_name='紧急联系人电话')
    family_address = models.CharField(max_length=200, blank=True,
null=True,verbose_name='家庭住址')
    now_address = models.CharField(max_length=200, blank=True,
null=True,verbose_name='现在住址')
    guarder = models.CharField(max_length=200, blank=True,
null=True,verbose_name='监护人')
    guarder_phone = models.CharField(max_length=200, blank=True,
null=True,verbose_name='监护人电话')
    description = models.CharField(max_length=200, blank=True,
null=True,verbose_name='备注')
    def __str__(self):
        return self.student_name
    class Meta:
        verbose_name_plural='学生表'
        # managed = False
        db_table = 'student'
#教室表
class Room(models.Model):
    id=models.BigAutoField(primary_key=True,blank=False)
    room_name = models.CharField(max_length=200,verbose_name='教室名')
    room_count = models.IntegerField(blank=True, null=True,verbose_name='教室容
量')
    room_status = models.IntegerField(blank=True, null=True,verbose_name='教室状
态',help_text='0教室空闲 1教室禁用',default=0)
    description = models.CharField(max_length=200, blank=True,
null=True,verbose_name='描述')
    def __str__(self):
        return self.room_name
    class Meta:
        verbose_name_plural='教室表'
        # managed = False
        db_table = 'rooms'
#课程表
class CourseWeekData(models.Model):
    id=models.BigAutoField(primary_key=True,blank=False)
    week = models.IntegerField(verbose_name="周数",default=0)
    Section=models.IntegerField(verbose_name="节次",null=False)
    subject=models.CharField(verbose_name="课程名", max_length=50,null=False)
    day=models.IntegerField(verbose_name="星期",null=False)
    #课程表和老师表、班级表、教室表都是一对多的关系
    teacher=models.ForeignKey(Teacher, verbose_name='老师',
on_delete=models.CASCADE)
```

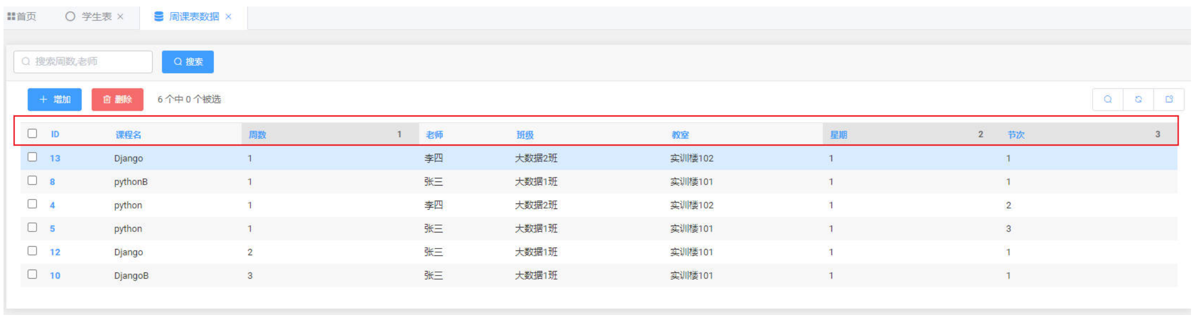
```
    Class=models.ForeignKey(BasicClass, verbose_name='班级',
on_delete=models.CASCADE)
    room=models.ForeignKey(Room, verbose_name='教室', on_delete=models.CASCADE)
    def __str__(self):
        return str(self.week)
    class Meta:
        verbose_name_plural='周课表数据'
        db_table='course_weekdata'
```

- **后台管理**

  后台管理使用的Django的第三方ui库simpleui来美化后台界面，参数配置如下。

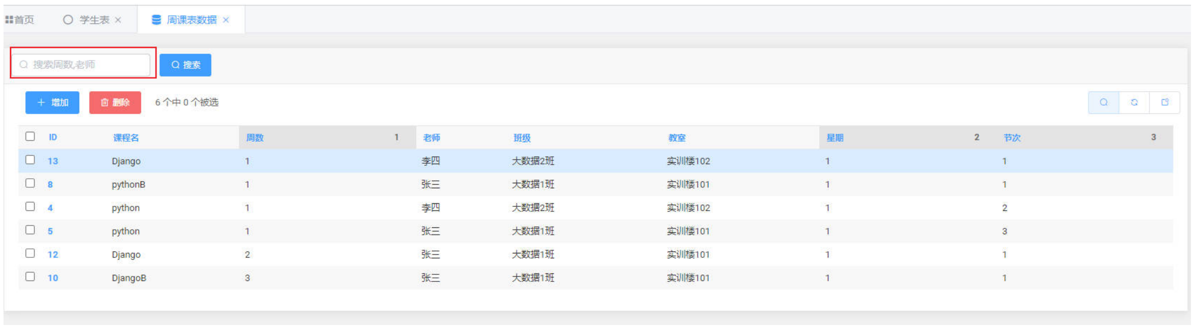
```
#settings.py
#后台logo
SIMPLEUI_LOGO=r'/static/favicon.ico'
#关闭simpleui广告
SIMPLEUI_HOME_INFO = False
#后台首页
SIMPLEUI_HOME_PAGE = '/root_index'


INSTALLED_APPS = [
    'simpleui',#注册simpleuiAPP，需要放到最前面
    'django.contrib.contenttypes',
    'django.contrib.admin',
    'django.contrib.auth',

    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'courseIndex',#注册courseIndexAPP
]
```

list_display是后台管理界面需要显示出的字段



| ID | 课程名 | 周数 1 | 老师 | 班级 | 教室 | 星期 2 | 节次 3 |
|---|---|---|---|---|---|---|---|
| 13 | Django | 1 | 李四 | 大数据2班 | 实训楼102 | 1 | 1 |
| 8 | pythonB | 1 | 张三 | 大数据1班 | 实训楼101 | 1 | 1 |
| 4 | python | 1 | 李四 | 大数据2班 | 实训楼102 | 1 | 2 |
| 5 | python | 1 | 张三 | 大数据1班 | 实训楼101 | 1 | 3 |
| 12 | Django | 2 | 张三 | 大数据1班 | 实训楼101 | 1 | 1 |
| 10 | DjangoB | 3 | 张三 | 大数据1班 | 实训楼101 | 1 | 1 |

search_fields是后台参与搜索的字段



| ID | 课程名 | 周数 1 | 老师 | 班级 | 教室 | 星期 2 | 节次 3 |
|---|---|---|---|---|---|---|---|
| 13 | Django | 1 | 李四 | 大数据2班 | 实训楼102 | 1 | 1 |
| 8 | pythonB | 1 | 张三 | 大数据1班 | 实训楼101 | 1 | 1 |
| 4 | python | 1 | 李四 | 大数据2班 | 实训楼102 | 1 | 2 |
| 5 | python | 1 | 张三 | 大数据1班 | 实训楼101 | 1 | 3 |
| 12 | Django | 2 | 张三 | 大数据1班 | 实训楼101 | 1 | 1 |
| 10 | DjangoB | 3 | 张三 | 大数据1班 | 实训楼101 | 1 | 1 |

ordering是参与排序的字段

```
#admin.py
from django.contrib import admin
# Register your models here.
from  .models import *

#后台名字
admin.site.site_title = "后台管理"
admin.site.site_header = "排课系统"

#注册老师表后台管理功能
@admin.register(Teacher)
class TeacherAdmin(admin.ModelAdmin):
    exclude = []
    list_display=['teacher_num','teacher_name']
    search_fields=['teacher_name']
#注册课程表后台管理功能
@admin.register(CourseWeekData)
class CourseWeekDataAdmin(admin.ModelAdmin):
    exclude = []
    list_display =
['id','subject','week','teacher','Class','room','day','Section']
    ordering=['week','day','Section']
    search_fields=['week','teacher']
#注册班级表后台管理功能
@admin.register(BasicClass)
class BasicClassAdmin(admin.ModelAdmin):
    exclude = []
    search_fields=['class_name']
#注册教室表后台管理功能
@admin.register(Room)
class RoomAdmin(admin.ModelAdmin):
    exclude = []
    search_fields=['room_name']
#注册学生表后台管理功能
@admin.register(Student)
class StudentAdmin(admin.ModelAdmin):
    list_display=['student_num','student_name']
    exclude = []
    search_fields=['student_name']
```

- **项目路由**

```
#course.urls.py
#主路由
from django.contrib import admin
from django.urls import path,include
from django.conf.urls.static import static
from django.conf import settings
urlpatterns = [
    #管理员后台
    path('admin/', admin.site.urls,name='admin'),
    #拼接courseIndex应用所有路由
    path('', include("courseIndex.urls")),
]+ static(settings.MEDIA_URL, document_root = settings.MEDIA_ROOT)

#courseIndex路由
#courseIndex.urls.py
from django.urls import path
from . import views
from django.conf.urls import url
app_name='courseIndex'

urlpatterns = [
    #管理员首页
    path('root_index', views.root_index,name='root_index'),
    #获取当前可带课的老师数据
    path('getteacher',views.getteacher,name='getteacher' ),
    #获取当前空闲教室数据
    path('getroom',views.getroom,name='getroom' ),
    #获取当前无课班级数据
    path('getclasses',views.getclasses,name='getclasses' ),
    #获取当前课表数据
    path('getdata',views.getdata,name='getdata' ),
    #添加课表详细接口
    path('adddata',views.adddata,name='adddata' ),
    #删除课表详细接口
    path('deldata',views.deldata,name='deldata' ),
    #保存当前课表数据
    path('savedata',views.savedata,name='savedata' ),
    #首页
    path('', views.index,name='index'),
    #登录页
    path('login', views.Login.as_view(),name='login'),

]
```

- **视图函数**

```
#courseIndex.views.py
#导包
from django.shortcuts import render, HttpResponse, redirect, reverse
from django.http import JsonResponse
from .models import *
import pandas as pd
import datetime
from django.db.models import Q
from django.views import View
from django.contrib.auth import login, logout
# 获取可带课教师
def getteacher(request):
    #获取前端由GET方式传入的参数(当前周数、当前星期、当前节次信息)
    week = int(request.GET.get('week', 1))#当前周数
    Section = int(request.GET.get('Section', 1))#当前节次
    day = int(request.GET.get('day', 1))#当前星期
    #如果为-1周，返回所有教师数据，返回格式为json,方便前端获取参数
    if week == -1:
        teachers = [{'id': i.id, 'name': i.teacher_name}
```

```
                      for i in Teacher.objects.all()]
        return JsonResponse({"list": teachers}, json_dumps_params=
{'ensure_ascii': False})
    #查找到当前时段有课的老师数据
    exclude_teachers = CourseWeekData.objects.filter(
        Q(week=week) & Q(day=day) & Q(Section=Section)).values_list('teacher')
    #筛选掉有课的老师，并返回json数据列表，格式为[{'id': i.id, 'name': i.teacher_name}]
    teachers = [{'id': i.id, 'name': i.teacher_name}
                for i in Teacher.objects.exclude(Q(pk__in=exclude_teachers))]
    return JsonResponse({"list": teachers}, json_dumps_params={'ensure_ascii':
False})
#获取空闲教室，原理同上
def getroom(request):
    week = int(request.GET.get('week', 1))
    Section = int(request.GET.get('Section', 1))
    day = int(request.GET.get('day', 1))
    exclude_rooms = CourseWeekData.objects.filter(
        Q(week=week) & Q(day=day) & Q(Section=Section)).values_list('room')
    rooms = [{'id': i.id, 'name': i.room_name}
             for i in Room.objects.exclude(Q(pk__in=exclude_rooms))]
    return JsonResponse({"list": rooms}, json_dumps_params={'ensure_ascii':
False})
#获取空闲班级，原理同上
def getclasses(request):
    week = int(request.GET.get('week', 1))
    Section = int(request.GET.get('Section', 1))
    day = int(request.GET.get('day', 1))
    if week == -1:
        classes = [{'id': i.id, 'name': i.class_name}
                   for i in BasicClass.objects.all()]
        return JsonResponse({"list": classes}, json_dumps_params=
{'ensure_ascii': False})
    exclude_classes = CourseWeekData.objects.filter(
        Q(week=week) & Q(day=day) & Q(Section=Section)).values_list('Class')
    classes = [{'id': i.id, 'name': i.class_name}
               for i in BasicClass.objects.exclude(Q(pk__in=exclude_classes))]
    return JsonResponse({"list": classes}, json_dumps_params={'ensure_ascii':
False})
#获取课程表数据
def getdata(request):
    #获取当前周次
    week = int(request.GET.get('week', 1))
    #筛选条件
    cid = request.GET.get('cid', '')#班级id
    tid = request.GET.get('tid', '')#老师id
    #初始化课程表
    '''
    [[[], [], [], [], [], []],
     [[], [], [], [], [], []],
     [[], [], [], [], [], []],
     [[], [], [], [], [], []],
     [[], [], [], [], [], []],
     [[], [], [], [], [], []],
     [[], [], [], [], [], []]]
    是一个7行6列的列表，行代表星期，列代表课程节次，最里面的小列表是为了处理一个时间段有多个班级
多门课的情况
    '''
    li = [[[] for j in range(6)] for i in range(7)]
    #如果传入了老师/班级筛选条件，就进行筛选
    if cid == '':
        courses = CourseWeekData.objects.filter(Q(week=week))
    else:
        courses = CourseWeekData.objects.filter(Q(week=week) & Q(Class_id=cid))
    if tid == '':
        pass
    else:
        courses = courses.filter(teacher_id=tid)
    #将每行里的小列表填上查到的课表数据
    for course in courses:
        li[course.day-1][course.Section-1].append({'id': course.id, 'subject':
course.subject, 'teacher': course.teacher.teacher_name, 't_id':
course.teacher.id, 'class': course.Class.class_name,
                                                  'c_id': course.Class.id,
'room': course.room.room_name, 'r_id': course.room.id, 'week': course.week,
'day': course.day, 'Section': course.Section})
    return JsonResponse({"list": li, 'week': week}, json_dumps_params=
{'ensure_ascii': False})

#保存csv，查询原理同上
def savedata(request):
    week = int(request.GET.get('week', 1))
    cid = request.GET.get('cid', '')
    cname = ''
    tid = request.GET.get('tid', '')
    li = {f'星期{i+1}': ['' for j in range(6)] for i in range(7)}
    if cid == '':
        courses = CourseWeekData.objects.filter(Q(week=week))
    else:
        cname = BasicClass.objects.get(id=cid).class_name
        courses = CourseWeekData.objects.filter(Q(week=week) & Q(Class_id=cid))
    if tid == '':
        pass
    else:
        courses = courses.filter(teacher_id=tid)
    for course in courses:
        li[f'星期{course.day}'][course.Section-1] += course.subject+'@' + \
            course.teacher.teacher_name+'@' + \
            course.Class.class_name+'@'+course.room.room_name+'\n'
    #将查询到的课表数据转换为DataFrame对象
    df = pd.DataFrame(li)
    #改字段名
    df.columns = ['星期一', '星期二', '星期三', '星期四', '星期五', '星期六', '星期天']
    df.index = [i for i in range(1, 7)]
    #生成课表名
    name = f'{cname}第{week}周课程表_{datetime.datetime.now().timestamp()}.csv'
    #保存至项目media/csv/目录下
    df.to_csv(r'media/csv/'+name)
    #返回保存路径
    return redirect('/media/csv/'+name)
```

```python
#添加课表数据
def adddata(request):
    #判断是否登录管理员账号，如果未登录返回管理员登录界面
    if request.user.is_authenticated:
        data = {}
        try:
            #获取前端传入的添加课表数据
            id = request.GET.get('id', '')
            week = int(request.GET.get('week', 1))
            Section = int(request.GET.get('Section', 1))
            day = int(request.GET.get('day', 1))
            subject = request.GET.get('subject', None)
            t_id = int(request.GET.get('t_id', ''))
            c_id = int(request.GET.get('c_id', ''))
            r_id = int(request.GET.get('r_id', ''))
            teacher = Teacher.objects.get(id=t_id)
            Class = BasicClass.objects.get(id=c_id)
            room = Room.objects.get(id=r_id)
            #保存至数据库
            if id == '':
                CourseWeekData.objects.update_or_create(
                    week=week,
                    Section=Section,
                    day=day,
                    subject=subject,
                    teacher=teacher,
                    Class=Class,
                    room=room
                )
            else:
                CourseWeekData.objects.filter(id=id).update(
                    week=week,
                    Section=Section,
                    day=day,
                    subject=subject,
                    teacher=teacher,
                    Class=Class,
                    room=room
                )
            data['status'] = 200
            data['msg'] = 'success'
        except Exception as e:
            data['msg'] = f'{e}'
            data['status'] = 500
        #返回该操作的状态和消息
        return JsonResponse({"data": data}, json_dumps_params={'ensure_ascii':
False}, status=data['status'])
    else:
        return redirect('/admin')

#删除课程表数据
def deldata(request):
    #判断是否登录管理员账号，如果未登录返回管理员登录界面
    if request.user.is_authenticated:
        data = {}
        try:
            #获取传入的id数据
            id = int(request.GET.get('id', ''))
            #数据库删除id相等的数据
            CourseWeekData.objects.get(id=id).delete()
            data['status'] = 200
            data['msg'] = 'success'
        except Exception as e:
            data['msg'] = f'{e}'
            data['status'] = 500
        #返回该操作的状态和消息
        return JsonResponse({"data": data}, json_dumps_params={'ensure_ascii':
False}, status=data['status'])
    else:
        return redirect('/admin')

#管理员首页界面
def root_index(request):
    #如果管理员未登录转入登录界面
    if str(request.user) != 'root':
        return redirect('/admin')
    #课表的查询数据
    week = int(request.GET.get('week', 1))
    cid = request.GET.get('cid', '')
    tid = request.GET.get('tid', '')

    if week <= 0:
        week = 1
    return render(request, 'root_index.html', {'week': week, 'cid': cid, 'tid':
tid})

#用户首页
def index(request):
    week = int(request.GET.get('week', 1))
    #获取cookie数据
    cid = request.COOKIES.get('cid', '')
    tid = request.COOKIES.get('tid', '')
    sid = request.COOKIES.get('sid', '')
    user=''
    if sid:
        user=Student.objects.get(id=sid)
    if tid:
        user=Teacher.objects.get(id=tid)

    if week <= 0:
        week = 1
    if user=='':
         return redirect('/login')
    if str(request.user) == 'root':
        return redirect('/root_index')
    return render(request, 'index.html', {'week': week, 'cid': cid, 'tid': tid,
'user': user})
```

```python
class Message:
    def __init__(self, status=None, msg=None):
        status = status
        msg = msg

    def __str__(self):
        return f'{self.msg}{self.status}'

#用户登录界面
class Login(View):
    def get(self, request):
        message = Message()
        logout(request)
        return render(request, 'login.html', {'message': message})

    def post(self, request):
        message = Message()
        no = request.POST.get('no', '')
        type = request.POST.get('type', '学生')
        pwd = request.POST.get('pwd')
        tid = ''
        cid = ''
        sid=''
        if no:
            try:
                if type == '学生':

                    u = Student.objects.get(student_num=no)
                    sid=no
                    cid = u.Class.id

                else:
                    u = Teacher.objects.get(teacher_num=no)
                    tid = no
            except Exception as e:
                message.status = 0
                message.msg = f'学号/工号错误'
                return render(request, 'login.html', {'message': message})
        else:
            message.status = 0
            message.msg = '请输入学号/工号'
            return render(request, 'login.html', {'message': message})
        if u.password != pwd:
            message.msg = '密码错误!'
            message.status = 0
            return render(request, 'login.html', {'message': message})
        else:
            r=redirect(f'/')
            r.set_cookie('tid',tid)
            r.set_cookie('cid',cid)
            r.set_cookie('sid',sid)
            return r
```

- **前端技术栈**

html,css,js,jquery.js、bootstrap

## 8.项目部署

- 创建虚拟环境

```
conda create -n course python=3.8
```

- 安装项目依赖

```
#进入项目虚拟环境
conda activate course
#进入项目目录
cd xxx
#安装依赖
pip install -r requirements.txt
```

- 修改数据库设置

```
#进入项目settings.py，修改数据库配置
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': '数据库名',
        'USER':'用户名',
        'PASSWORD':'密码',
        'HOST':'127.0.0.1',
        'POST':'3306',
        'OPTIONS': {'charset': 'utf8mb4'},
    }
}

```

- 迁移数据库

```
#进入项目目录
cd xxx
python manage.py makemigrations
python manage.py migrate
```

- 启动项目

```
python manage.py runserver
```